Software Verification
Exercise Session 1 Solution

We present proof in outline form - you can also use explicit lists of theorems or proof trees.

• 9.3

\{x = a \land y = b\}
\{x+y = a+b \land x = a\}
\quad t := x
\{x+y = a+b \land t = a\}
\quad x := x + y
\{x = a+b \land t = a\}
\quad y := t
\{x = a+b \land y = a\}

• 9.6

1)
\{z*x^y = K\}
\{(z*x)*x^{y-1} = K\}
\quad z := z*x
\{z*x^{y-1} = K\}

2)
\{z*x^y = K\}
\{(z*x)*x^{y-1} = K\}
\quad y := y-1
\{z*x^{y-1} = K\}
\quad z := z*x
\{z*x^y = K\}

3)
\{y \text{ even } \land z*x^y = K\} // With integer arithmetic, we cannot assume 2(y/2) = y for all y.
\{z*(x^2)^{y/2} = K\}
\quad y := y/2
\{z*(x^2)^y = K\}
\quad x := x^2
\{z*x^y = K\}
4) Here is the inference rule for guarded commands of the form \texttt{if...}[]\, g_i : c_i [] ... \texttt{end}:
\[ P \Rightarrow (\forall i \in 1..n. \ g_i) \forall i \in 1..n. \ \{g_i \land P\}c_i\{Q\} \]

\[ \{P\} \texttt{if...}[]\, g_i : c_i [] ... \texttt{end} \{Q\} \]

Notice that the following implications hold (i.e. they are valid/tautologies):
\begin{enumerate}
  \item (\(z^x y = K\)) \Rightarrow (y \text{ odd} \lor y \text{ even}), and
  \item (y \text{ odd} \land z^x y = K) \Rightarrow (z^x y = K),
\end{enumerate}

Now we can apply the rule of Consequence with the triple from part 2 and the valid implication ii to obtain the triple:
\[ \{y \text{ odd} \land z^x y = K\} \quad y := y - 1 ; \quad z := z^x \quad \{z^x y = K\} \]

This triple, the triple from part 3 and the valid implication i fulfill all the premises of the rule. We can therefore infer the triple:
\[ \{z^x y = K\} \quad \text{if y odd} : y := y - 1 ; \quad z := z^x \quad \{z^x y = K\} \]

In proof outline form:
\[ \{z^x y = K\} \quad \text{// Remember that here is an implicit implication of the \lor of the guards!} \]
\begin{verbatim}
  if
    y odd :
      \{y odd \land z^x y = K\}
      \{z^x y = K\}
      \{(z^x)^y = K\}
      y := y - 1
      \{(z^x)^y = K\}
      z := z^x
      \{z^x = K\}
      []
    y even :
      \{y even \land z^x y = K\}
      \{z^x (x^{y/2}) = K\}
      y := y/2
      \{z^x (x^{y/2}) = K\}
      x := x^2
      \{z^x y = K\}
  end
\end{verbatim}

\[ \{z^x y = K\} \]

9.7

Recall the proof rule for \texttt{from..until} commands, where I is the loop invariant:
\[ \{P\} c_1 \{I\} \quad \{I \land \neg b\} c_2 \{I\} \]

\[ \{P\} \texttt{from} c_1 \texttt{until} b \texttt{loop} c_2 \texttt{end} \{I \land b\} \]

It should be clear that \(z^x y = K\) is an invariant of the loop.
With the usual backward assertion propagation, we can easily prove the initialization triple \( \{ m^n = K \} \ x := m \ ; \ y := n \ ; \ z := 1 \ \{ z^x y = K \} \).

By the rule of Consequence and the triple from 9.6.4, we also know:
\( \{ z^x y = K \wedge \neg(y=0) \} \text{ if } y \text{ odd} : y := y-1 ; z := z^x y \text{ else } y := y/2 ; x := x^2 \text{ end } \{ z^x y = K \} \).

Hence \( \{ m^n = K \} \text{ from...end } \{ z^x y = K \wedge y = 0 \} \) by the inference rule above, and with another application of Consequence, we know:
\( \{ m^n = K \} \text{ from...end } \{ z = K \} \)

Now since the \text{from...end} command did not modify \( m \), \( n \) or \( K \), we know that \( m^n = K \) still holds afterwards. Formally, we can apply the rule of Constancy:

\[
\begin{align*}
\{ P \} c \{ Q \} \\
\{ P \wedge R \} c \{ Q \wedge R \}
\end{align*}
\]

provided \( c \) does not modify (i.e. assign to) any of the free variables of \( R \).

In this case, the \( R \) will be \( m^n = K \), so we know:
\( \{ m^n = K \wedge m^n = K \} \text{ from...end } \{ z = K \wedge m^n = K \} \)

By the rule of Consequence, we again simplify and get:
\( \{ m^n = K \} \text{ from...end } \{ z = m^n \} \)

Next, we can apply the Auxiliary Variable Elimination rule to get rid of \( K \). The rule is:

\[
\begin{align*}
\{ P \} c \{ Q \} \\
\{ \exists v. P \} c \{ \exists v. Q \}
\end{align*}
\]

provided \( v \) does not occur free in \( c \).

So now we have \( \{ \exists K. m^n = K \} \text{ from...end } \{ \exists K. z = m^n \} \}, and we can simplify it with the rule of Consequence to get:
\( \{ \text{true} \} \text{ from...end } \{ z = m^n \} \)

We can now strengthen the precondition with the rule of Consequence to get:
\( \{ m>0 \wedge n\geq 0 \} \text{ from...end } \{ z = m^n \} \)

Hence, we have proven that the program computes \( m^n \) and stores the result in the variable \( z \). The \( n\geq 0 \) is important only for termination, which we have not proven.

Note: in a proof outline, an application of Constancy or Auxiliary Variable Elimination will be denoted by a level of indentation. For example, the application of Constancy above would be written:
\( \{ m^n = K \wedge m^n = K \} \)
\( \{ m^n = K \} \text{ from...end } \{ z = K \} \)
\( \{ z = K \wedge m^n = K \} \)
9.9

One can imagine several sound axioms of various strength. However, the following one is known to be equivalent to the well-known backward rule \{P[e/x]\}x := e\{P\}:

\{P\}x := e\{\exists x'. P[x'/x] \land x = e[x'/x]\}, where \(x'\) is fresh, i.e. it does not occur free in \(P\) or \(e\), and it is not the same variable as \(x\).

In the postcondition, the variable \(x'\) can be understood as recording what \(x\) used to be. So we can read the triple informally as: after executing \(x := e\), we remember that there used to be something (let's call it \(x'\)) such that \(P[x'/x]\) holds. Furthermore, the value of \(x\) is now updated to \(e\) where we are careful to replace occurrences of \(x\) in \(e\) by its old value \(x'\).

9.14

\textbf{repeat s until} \quad b = s ; \textbf{while } \neg b \textbf{ do } s \textbf{ end}

So we can propose the rule:

\[
\begin{align*}
\{P\}S\{I\} & \quad \{I \land \neg b\}S\{I\} \\
\hline
\{P\}\textbf{repeat s until} b\{I \land b\}
\end{align*}
\]

To see that the rule is sound (i.e. correct), notice that we can derive it as follows:

\[
\begin{align*}
\{I \land \neg b\}S\{I\} & \\
\hline
\textbf{While} \quad \{I\}\textbf{while } \neg b \textbf{ do } s\textbf{ end}\{I \land \neg b\} \\
\hline
\textbf{Consequence} \quad \{P\}S\{I\} & \quad \{I\}\textbf{while } \neg b \textbf{ do } s\textbf{ end}\{I \land b\} \\
\hline
\textbf{SequentialComposition} \quad \{P\}s ; \textbf{while } \neg b \textbf{ do } s \textbf{ end}\{I \land b\}
\end{align*}
\]